Guiding Wind Farm Optimization with Machine Learning

Bovet, Charles
bovet@stanford.edu

Iglesias, Ramon
rdit@stanford.edu

December 13, 2013

1 Introduction

1.1 Motivation

Wind farm micro-siting is a complex process that requires multidisciplinary analysis and optimization. The problem consists not only of positioning the wind turbines in the locations within the site that have the most wind energy potential, but also it has to minimize wake interference between turbines. This means that usually the sum of the expected isolated turbines is less than the actual performance. Additionally, cost of turbine installation varies with the terrain and there is significant cost of road and collection system installation. Additionally, wind farms are increasingly large and complex (constraint wise) making inefficient optimization algorithms impractical. All these different parameters add up to the complexity of micro-siting wind farms, however, the performance of the wind farm will largely depend on it.

1.2 Background

Current industry practice uses GIS based software to estimate the energy capture given turbine siting and meteorological conditions. An example of this is OpenWind by AWS Truepower. OpenWind uses GIS file formats to develop a model and simulate wind farm performance including wake effects. Additionally, it offers an optimization module that is mostly based on Gaussian movements of the individual turbines.; however, the optimization is too inefficient for large wind farms (on the order of 1000's of turbines).
On the other hand, academics have tackled the problem with a variety of approaches from standard Genetic Algorithms (GA), Particle Swarm Optimization (PSO) to Covariance Matrix Adaptation Based Evolutionary Estrategy (CMA-ES) and Morphogenetic Engineering (ME) [1]. In general, these approaches tend to either find excellent solutions but are highly costly computation wise or find decent solutions with decent (OpenWind) or excellent (ME) computation

costs. Additionally, with the exception of the OpenWind optimization, most studies run on simplified models.

1.3 Objective

Thus, our objective is to improve current optimization practices with machine learning algorithms. Our hypothesis states that with the tremendous amount of iterations that are run through evolutionary algorithms, the data produced should be useful for doing better decisions when updating our candidate solutions. In this sense, there are two approaches that can be taken. First, we can run an off-the-shelf GA and tweak it such that as more and more information is available, the subsequent populations are created based on a learning algorithm, similar to the Learnable Evolution Model (LEM) [2]. The second approach, is to take the Morphogenetic Engineering approach and train an agent (representing an individual turbine) to populate a given terrain after several iterations. However, because our intent is to be able to use OpenWind simulation capabilities to estimate the fitness function and OpenWind currently does not support variable number of turbines when used externally, we will stick to the first approach.

2 Methodology

2.1 Input and Output data

Different to most machine learning approaches, our problem setup requires us to generate our data in the fly. We will achieve this by using OpenWind as the fitness evaluation, but running the optimization algorithm outside of it. In this setup, the algorithm produces several candidates with specific feature values (the positions of the turbines) and receives from OpenWind a fitness value for each of them that the algorithm uses to learn. There is precedent of algorithms that attempt to tackle optimization problems in a similar fashion. In particular, a specific area of evolutionary algorithms called Estimation of Distribution Algorithms [3] use the same update/generate loop to guide the optimization. Using this background study and our knowledge of machine learning, we decided that the problem at hand closely resembled the spam-filter problem. In our case, a discretized grid of (x, y) points can be expressed as a binomial vector v with each of its entries representing a specific combination of x and y and the values of it's entries being 1 if it contains a turbine and 0 if it does not. Once we obtain the performance of each set of positions, we rank them and take top 50% as "high performing". Thus, each sample is given a value of $w = 1$ if they are on the top five, and 0 otherwise. Using this two transformations, we have our sample data $(v^{(i)}, w^{(i)})$ for each loop that will feed our learning algorithm. Given the level of similarity with the spam-filter problem, we decided to use Naive Bayes as our learning algorithm. Finally, once our algorithm has learned the parameters, it uses that same distribution to generate λ new individuals that are fed to the OpenWind instances.

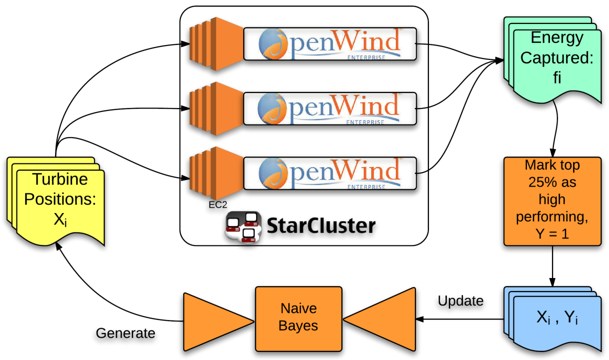

Figure 1: Summary of the optimization loop

2.2 Results and Discussion

After running the optimization for 3000 generations we got a decent overview of our performance. The following paragraphs summarize our findings.

Solutions and performance As seen in Figure 2, the solution improved rapidly on the early iterations (as it is in most optimization processes), however, it suffered of diminishing returns. It is interesting to note, however, that the mentioned figure presents the results as a fraction of the distance between the theoretical maximum (when all turbines absorb the full potential) and an approximated minimum (where all the turbines are clustered together as close as possible), in that context we see that our algorithm finds a very good solution, similar to the fractions presented at Wilson et al [1] (82% for the GA), but at a much higher level of computational expense (900K simulations vs 200K).

Average Error As a sanity check to see the validity of our algorithm, we estimated the average error of the Naive Bayes on each generation via cross-validation. As can be noticed in Figure 3, the average error of each generation oscillates around 10%, being much higher than what we had anticipated, we believe that perhaps using algorithms that take the structure of the relationship of the features in to account could be more effective.

Final Distribution As an additional exploratory endeavour, we compared the final distribution of the parameters and the performance of the turbines in the best solution. We can see in Figure 4 first that, as expected, the best solution has turbines where the distribution expects them to be and additionally,

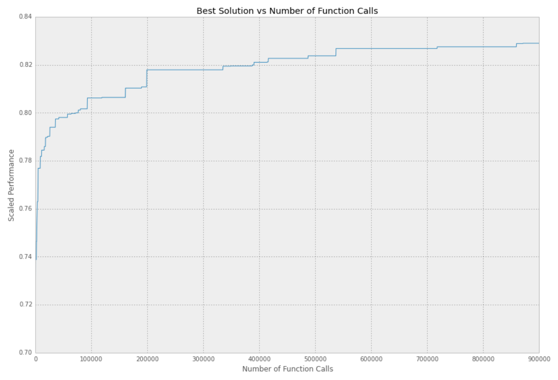

Figure 2: Improvement on the best solution.

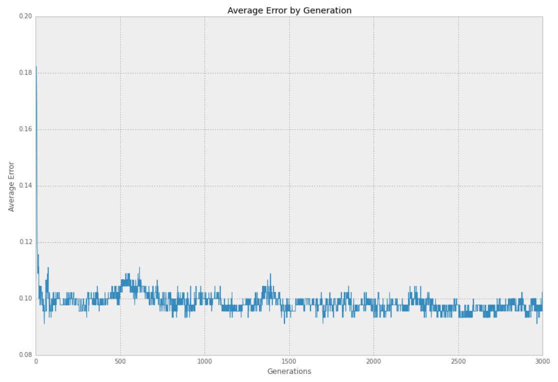

Figure 3: Average error by generation.

the performance of the turbines are generally higher where the probability of being selected is higher.

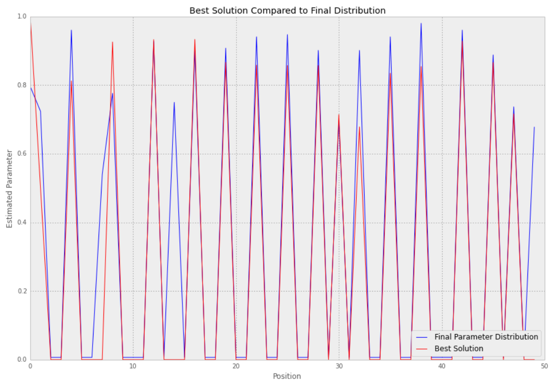

Figure 4: Final distribution of the parameters compared to the best solution. Note that the vertical axis represents performance for the best solution and probability for the distribution.

3 Conclusion

While the optimization did work, the performance is no where near to our expectations. To achieve similar results as in previous papers we needed much more simulations that are not feasible for our intentions. We believe that a big improvement could be done by using a structure inferring model (like Bayesian Optimization Algorithm) that can detect the subtle influence that each turbine has on the rest. An additional step could be using the performance of the individual turbines as well as the whole. In this fashion, we could probably give preference to higher performing turbines over just turbines that tend to appear in multiple good solutions. Finally, our method used

References

[1] Wilson, Dennis; Awa, Emmanuel; Cussat-Blanc, Sylvain; Veeramachaneni, Kalyan; O'Reilly, Una-May. *On Learning to Generate Wind Farm Layouts.* Proceeding of the fifteenth annual conferenc on Genetic and Evolutionary Computation Conference. Pages 767-774. ACM New York, NY. 2013.

[2] Michalsk, Ryszard S. *Learnable Evolution Model: Evolutionary Processes Guided by Machine Learning* Machine Learning. Pages 9-40. ACM. 2013.

[3] Larranaga, Pedro. *A review on estimation of distribution algorithms.* In Estimation of distribution algorithms, pp. 57-100.. ACM. I Springer US, 2002.